Twitter Sentiment Analyzer Code Documentation

Introduction:

This is a program that uses sklearn niave_bayes classifier, BernoulliNB, to analyze tweets from twitter, and classify them based on their polarity. The user puts in a keyword, and the output would be the sentiment score and the most recent 100 tweets with their labels.

Program Code Files:

This program is written in python programming language, and it was run using a jupyter notebook. Therefore the program consists of two jupyter Notebook files.

File 1: Twitter Sentiment Analyzer Trainer Code.

File 2: Twitter Sentiment Analyzer Program.

Running the code:

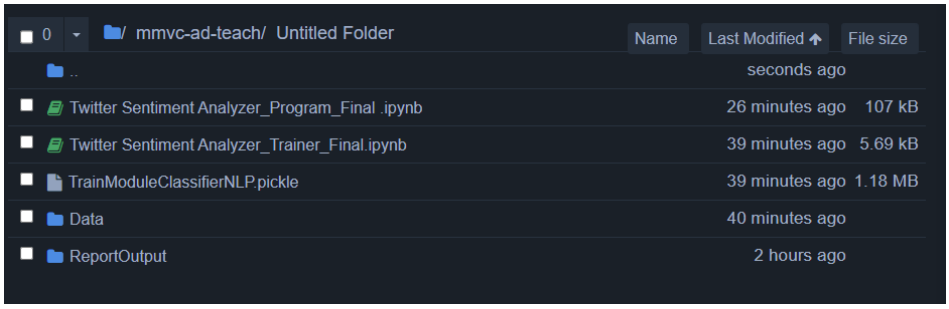
In order to run the code, you should have a **jupyter notebook** installed on your local machine. This code might require high cpu, In our case we used the Applied Machine Learning Class(course that we are taking) Server for fast performance, the code could also be ran on normal jupyter too but might take little bit more time.

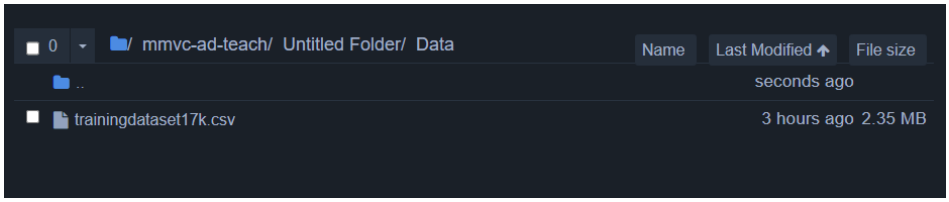
Follow the steps in order to run the code:

1. Make a new file in jupyter notebook.

   Upload Twitter Sentiment:

   1. *Twitter Sentiment Analyzer_Program_Final.ipynb*
   2. *Twitter Sentiment Analyzer_Trainer_Final.ipynb*
   3. *TrainModuleClassifierNLP.pickle.*
   4. Make an new folder called *Data* and upload *trainingdataset17k.csv* in it
   5. Make a new folder called *ReportOutput*





2. Open the notebook *Twitter Sentiment Analyzer_Program_Final.ipynb* and run it. *Note: Since we already **saved** the training Module from *Twitter Sentiment*

Analyzer_Trainer_Final.ipynb as ***TrainModuleClassifierNLP.pickle,*** **we won't need to run the** *Twitter Sentiment Analyzer_Trainer_Final.ipynb.*

Program Data Sets:

1. Training Data Set
    - The data set that was used is from kaggle . This data set consists of 1.6 million tweets, that are classified based on their polarity, where 0 is negative and 4 is positive. Out of the 1.6 million tweets, we chose random 17,000 tweets. 8500 were positive and 8500 were negative. We took those 17,000 tweets and imported them into a new csv file called training 'trainingdataset17k.csv'. Those 17, 000 tweets are the tweets that we will **train** in order to use the train module later on the tweets that we get from the server based on the user keyword.
2. Test Set
    - The test consisted of the data that were retrieved from twitter database after connecting to twitter. The data in the Test set consist of 100 recent tweets, that are retrieved from the server based on the user keyWord. This includes the tweet text, the user tweet and the time it was created at. Those 100 tweets are the tweets that we will find the labels for using the training module that we made out of the training data set, and then we will produce the sentiment score from.

Libraries:

1. Twitter Library: used to connect to twitter server.
2. Nltk: toolkit that is used for dealing with natural language
    a. SklearnClassifier: classifier used to build the training module
    b. Word Tokenize: tool used to tokenize the words, remove extra unnecessary letters from words. Used in the preprocessing class
    c. Stopwords: tool used to remove stopwords. Used in the preprocessing class
    d. Punkt: tokenizer that divides the string into a list of sentences
3. Sklearn naive_bayes BernoulliNB: classifier used for labeling the testing data using the train module.
4. Numpy: library used for working with arrays
5. Pandas: library used for data manipulation and analysis
6. Time: used to convert the twitter time into a string time.
7. re : used to remove unnecessary characters.
8. Strings punctiotation: returns all sets of punctuation.

Code Explained:

1. Twitter Sentiment Analyzer Trainer Code: Code Flow and Explanation:
   This code is to build the training module:
   a. Import Training Data:
      i. Importing the data from csv using pandas dataframe, we define the polarity in the data set where negative polarity is 0 while positive polarity is 4, we label the tweets based on their polarity, and at the end we convert the pandas dataframe to a dictionary.
   b. Preprocess The training data
      i. Here we clean up the data, so we basically take what we need for the sentiment analysis, therefore we remove URLs, usernames, and hashtags.
         1. Tweets Processing: loops inside the tweetlists, and calls tweet cleaner function to clean each tweet.
         2. Tweet cleaner function converts the text to low cases, then removes url, usernames, and hashtags, and then tokenize the word and then return.
   c. Build NLTKProccessor1 and NLTKProccessor2 :

      i. Using NLTKProccessor1 function to Build Vocabulary for NLTK using the preprocessed data.
         1. After we preprocess the data we use this function to build a vocabulary set.
         2. In order to do so: we get the words we need from the preprocessed training data and then we use nltk library to make sure that each word has its frequency and then save those frequencies as keys.
      ii. Using NLTKProccessor2 function to extract features from the build vocabulary list.
         1. Function to check whether the words from the vocabulary are in the tweets, we use json library and based on that it returns true or false.
   d. Make a training feature vector using NLTKProccessor1 and NLTKProccessor2.
   e. Make a training module using SklearnClassifier BernoulliNB, and we train the feature vector.
   f. Save the training module so we won't have to keep repeating line 6,and we use it in the Twitter Sentiment Analyzer Trainer Program

2. Twitter Sentiment Analyzer Trainer Program: Code Flow and Explanation:
   This is the main code for the program

a. Import Training Data: just like step A in the training code.
b. Configure twitter api
    i. We use the credentials that we got from twitter developer after being approved to access the database
c. Import Test Set:
    i. We use twitter api in order to do so, so we configure
d. User Enters an input:
    i. It checks if the input is space or no input and ask him to do so again
e. Building The Test Set:
    i. We use the key word given by the user to retrieve tweets.
    ii. We use the function GetSeaarch from twitter api, we pass in the function the keyword and we set the arguments of count to 100 because that's the most u can retrieve with this function, lang en inorder to retrieve english tweets, and the result type to recent to get recent tweets.
    iii. Then we append all of the tweets to an array.
f. Preprocess the data: Same as step B in the training code.
    i. In this case we preprocess both the training set and the testing set
g. Build NLTKProccessor:
    i. Using NLTKProccessor function to Build Vocabulary for NLTK using the preprocessed data: Same as step C-i in the training code.
    ii. Using NLTKProccessor function to extract features from the build vocabulary list. : Same as step C-ii in the training code.
h. Load the training module build in the training code
i. Use the training module to classify the tweets in the testing data.
    i. We use the NLTK feature processor that we build on each tweet text.
    ii. This classify our 100 tweets into positive or negative
    iii. This return a list of Labels "positive" and "negative"
j. Generate Score and Report:
    i. Generating sentiment score by counting how many positive and negative labels returned from the classifier(j).
    ii. If positive count is more than negative count: sentiment would be positive, and Sentiment Score would be positive count / the length of the list that contains the labels for each tweet. And same for if negative count is bigger than positive count.
    iii. We write the findings on a txt file(Number of tweets retrieved, Number of positive tweets, number of negative tweets. This is part of the report.
    iv. Then we use pandas to do a table where we would have the tweets and their labels next to them for visualization.
    v. We export the table into an excel file, as this is the second part of the report.

Useful Links that helped in the implementation:

1. https://www.kaggle.com/kazanova/sentiment140
2. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html
3. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
4. https://pythonprogramming.net/sklearn-scikit-learn-nltk-tutorial/
5. https://www.nltk.org/howto/classify.html
6. https://towardsdatascience.com/beginners-guide-for-data-cleaning-and-feature-extraction-in-nlp-756f311d8083
7. https://monkeylearn.com/blog/sentiment-analysis-of-twitter/
8. https://www.pluralsight.com/guides/building-a-twitter-sentiment-analysis-in-python
9. https://www.tweetbinder.com/blog/twitter-sentiment-analysis/
10. https://towardsdatascience.com/creating-the-twitter-sentiment-analysis-program-in-python-with-naive-bayes-classification-672e5589a7ed
11. https://lionbridge.ai/articles/how-to-build-a-twitter-sentiment-analysis-system/
12. https://www.pluralsight.com/guides/building-a-twitter-sentiment-analysis-in-python
13. https://medium.com/better-programming/twitter-sentiment-analysis-15d8892c0082